Technical breakdown of position based fluid

CHANGLIN SU* and QINGYUAN QIE*, University of Toronto, Canada

This paper will implement a parelle version of Position Based Fluids (PBF) method [Miles Macklin et al. 2013] for real-time fluid simluation and disscuss the effects of different confinement in the original paper. Compare to BPF, the widly used Lagrangian methods often secrafies the incompressablity of fluids for better performance, which cause the reuslt to be inaccurate compare to the behavior of real fluids. To addess this isssue, PBF method numerically integrate quatities from the neibouring particles with a smoothing kernel to calculate displacement while enforcing the incompressable contraint of fluid for the Navier-Strokes equations. This paper uses CUDA framework in our implementation to achieve real-time, accurate result.

CCS Concepts: • **Computer methodologies** → **Pysical simulation.**.

Additional Key Words and Phrases: Fluids, Physically Based Animation

ACM Reference Format:
Changlin Su and Qingyuan Qie. 2018. Technical breakdown of position based fluid. *ACM Trans. Graph.* 37, 4, Article 111 (August 2018), 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Fluids such as water or gases, has played a important role in the field of computer based physical animations. Although fluid may look simple, but they are much more complicate to simulate than solids. The motion of fluids are governed by the Navier–Stokes equations. The solutions of the Navier–Stokes equations often include turbulence which makes finding a analytical solution exteremly hard in three dimensions. Although countless efforts are put into this field, the progress is still limited. Currently numerical simluation remains the most effective way to approximate a complex Navier–Stokes systems. In this report, we'll breakdown the Position based method and disscuss the significans and effects of all the constrant it proposed. In positioned based fluid, fluids are divided into particles and then compute the interactions between the fluid particles such as density and viscosity confinement, combine with external forces, we then update the velocities and positions of the fluid particle. To optimize for better performance, we divided the simulation spaces into grids so that particles only interact with other particles in the neibouring grid.

*Both authors contributed equally to this research.

Authors' address: Changlin Su, sheldon.su@mail.utoronto.ca; Qingyuan Qie, placeholder, University of Toronto, Toronto, Ontario, Canada.

0730-0301/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

2 RELATE WORK

According to simulation method, algorithm can generally be divide to two different catgories: Euler and Lagrangian methods. Euler methods split the space into grids and uses fintie differential mehtod to evualate the model. One of the example is Stam's stable fluid [Stam 1999]. The Lagrangian methods divide the fluid into free particles under constrains. One of the most known method of this kind is the SPH method [Smoothed Particle Hydrodynamics]. Both category has its own drawbacks: Euler methods easyily effected by numerical dissapation during simulation and Lagrangian methods needs smaller steps each iteration to ensure the stability of the algorithm. Current fluid simulation methods such as Position Based Fluid mehtods uses a hybrid of both method to achieve high level of accuracy while having reasonable performance.

3 PBF METHOD

For any Abitary fluid, its motion is governed by the Navier–Stokes equation:

$$\rho \frac{\partial \boldsymbol{v}}{\partial t} = \rho \boldsymbol{g} - \nabla p + \mu \nabla^2 \boldsymbol{v} \tag{1}$$

where ρ is the density of the fluid, μ is the viscosity constant of the fluid, p is the pressure and v denote the velocity. However, the above equation does not satisfy the incompressable property of fluids, so an additional constrains needs to be added:

$$\nabla \cdot \boldsymbol{v} = 0 \tag{2}$$

In PBF method, we represent the fluid with N particels, and their postions is denote as $\boldsymbol{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_2\}$.

3.1 Density confinement

Since we are simulating a fluid, we have to satisfy the incompressable property of the fluid. Instead of statisfying equation (2) directly, BPF uses desity constraint to satisfy the reqirement:

$$C_i(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_2) = \frac{\rho_{\mathrm{i}}}{\rho_0} - 1 \tag{3}$$

Where ρ_0 is the rest density of the fluid and ρ_{i} is given by the standard SPH estimator:

$$\rho_{\mathrm{i}} = \sum_j m_j W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{4}$$

where m_j is the mass of the neiboring particles and W is the poly6 smoothing kernel as gievn in [Müller et al. 2003]. In this paper, we assume all the particles have a mass of 1. Since a fluid is incompressable, intuitively this means that the density for any given part of the fluid should equal to the rest density of the fluid, which means the result of (3) should be equal to zero at all time:

$$C_i(\boldsymbol{p}_1, \boldsymbol{p}_2, ..., \boldsymbol{p}_2) = 0 \tag{5}$$

Furthurmore, when there is an displacement in the fluid particle, the density should also reamins the same, this property is statisfied through the following constraint:

$$C(\boldsymbol{p} + \Delta\boldsymbol{p}) = 0 \tag{6}$$

We can estimate a proper $d\Delta p$ equation (6) with Newton steps along the gradent of equation(5):

$$\Delta p = \nabla C(\boldsymbol{p})\lambda \tag{7}$$

And we can approximate the value of equation (6) with linear approximation:

$$C(\boldsymbol{p} + \Delta\boldsymbol{p}) \approx C(\boldsymbol{p}) + \nabla C^T \nabla C \lambda \tag{8}$$

Now we only need the value of ∇C^T to estimate a proper λ. Luckly, ∇C^T can be defined as follows:

$$\nabla_{\boldsymbol{p}_k} C_i = \begin{cases} \sum_j \nabla_{\boldsymbol{p}_k} W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) & \text{if } k = i \\ -\nabla_{\boldsymbol{p}_k} W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) & \text{if } k = j \end{cases} \tag{9}$$

Plugging this back to (8) and solve for λ:

$$\lambda_\mathrm{i} = \frac{\nabla_{\boldsymbol{p}_k} C_i}{\sum_k |\nabla_{\boldsymbol{p}_k} C_i|^2 + \epsilon} \tag{10}$$

Note that when there are not particles nearby, $\sum_k |\nabla_{\boldsymbol{p}_k} C_i|^2$ might be zero, so a extra ϵ term is added to prevent the denominator from evaluating to zero.

3.2 Tensile Instability

The original SPH algorithm has an issue known as particle clamping, where particles clamps to each other. Therefore, a corrective term s_{corr} is added when computing $\Delta\boldsymbol{p}$:

$$\Delta\boldsymbol{p}_i = \frac{1}{\rho_0} \sum_j (\lambda_\mathrm{i} + \lambda_\mathrm{j} + s_{corr}) \nabla W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{11}$$

And it is define as follows:

$$s_{corr} = \left(\frac{-kW(\boldsymbol{p}_i - \boldsymbol{p}_j, h)}{W(\Delta\boldsymbol{q}, h)} \right)^n \tag{12}$$

Where k, n, h and Δq is selected constants. In our implementation, we use $k = 0.1$, $n = 4$, $h = 0.1$ and $|\Delta q| = 0.1h$. With this corrective term, PBF does not required 30-40 neiboring particles at all time like SPH does, improcing simulation efficency.

3.3 Vorticity and viscosity confinement

Since PBF introduced addtional damping which is undesireable, [Fedkiw et al. 2001] introduced vorticity confinement which has the following form:

$$\mathbf{f}_i^{\text{vorticity}} = \epsilon(\mathbf{N} \times \omega_\mathrm{i}) \tag{13}$$

where

$$\omega_\mathrm{i} = \sum_j (\mathbf{v}_j - \mathbf{v}_i) \times \nabla_{\mathbf{p}_j} W(\mathbf{p}_i - \mathbf{p}_j, h) \tag{14}$$

And $\mathbf{N} = \frac{\omega_\mathrm{i}}{\nabla|\omega|_i}$.

Note that we did not implement this vorticity confinement in our code.

Compare to vorticity confinement, viscosity confinement is relatively simple. We apply the XSPH viscosity estimator to oue velocity update from [Schechter and Bridson 2012] to accomadate the coherent motion for any arbitary fluid:

$$\mathbf{v}_i^{new} = \mathbf{v}_i + c \sum_j (\mathbf{v}_j - \mathbf{v}_i) W(\boldsymbol{p}_i - \boldsymbol{p}_j, h) \tag{15}$$

Equation (15) will introduce addtional interation between neiboring particles so the movements of neiboring particle tend to "drag" the target particle to move with them.

4 RESULTS

We implemented the PBF method using CUDA framework. Note that in our implementation we omited the vorticity confinement due to time limits. We simulated a sene where a block of water falls into a empty square space. Our simulation has 8000 particles in total and wer are able to render their movements in real time. The behavior of our simulated fluid agree with our expectation: The fluid particles having coherent motion while maintaing adquate distance with each other (less compression). Therefore we concluded that our simulation produces a accurate result.

5 DISCUSSION

In this paper, we covered the technical details of the PBF and provided a parallel implmentation using CUDA framework and a optimization inspired by [TODO: NVDIA and tongji]. BPF method produces a realistic simulation to the human eye while maintaining a resonable performance. One limit of our implementation is that vorticity confinment is not implemented, which will cause the fluid to be more damped than in reality. In our implementation,we observed a performance drop of the simulation when the fluid particles start hitting the boundry. We conclude that this is due to particles concentrated in a few grid cells in space, hence more computation needed for each cells.

6 REFERENCES